

Data-Driven Learning and Planning for Environmental Sampling

Kai-Chieh Ma, Lantao Liu, Hordur K. Heidarsson, Gaurav S. Sukhatme

Department of Computer Science

University of Southern California

Los Angeles, CA 90089

{kaichiem, lantao.liu, heidarss, gaurav}@usc.edu

Abstract

Robots such as autonomous underwater vehicles (AUVs) and autonomous surface vehicles (ASVs) have been used for sensing and monitoring aquatic environments such as oceans and lakes. Environmental sampling is a challenging task because the environmental attributes to be observed can vary both spatially and temporally, and the target environment is usually a large and continuous domain whereas the sampling data is typically sparse and limited. The challenges require that the sampling method must be informative and efficient enough to catch up with the environmental dynamics. In this paper we present a planning and learning method that enables a sampling robot to perform persistent monitoring tasks by learning and refining a spatiotemporal environmental model. Our environmental sampling framework consists of two components: to maximize the information collected, we propose an informative planning component that efficiently generates sampling waypoints that contain the maximal information; To alleviate the computational bottleneck caused by large-scale data accumulated, we develop a component based on a sparse Gaussian Process whose hyperparameters are learned online by taking advantage of only a subset of data that provides the greatest contribution. We validate our method with both simulations running on real ocean data and field trials with an ASV in a lake environment. Our experiments show that the proposed framework is both accurate and efficient in learning the spatiotemporal environmental model[†].

1 Introduction

Scientists are able to gain a greater understanding of the environmental states (e.g., physical, chemical or biological parameters) through environmental sensing and monitoring [Dunbabin and Marques, 2012]. Typically, the environmental sensing involves a process of collecting data of important environmental attributes (e.g., temperature, salinity, pollution contents) at certain selected locations, and the goal is to build a “data map” that can best describe the state of the environment.

However, the environment to be monitored is usually a large and continuous area whereas the sampled data is discrete and limited due to cost. In addition, the outdoor water environment is typically dynamic, so that any environmental attribute associated to water also varies as time elapses. Fig. 1 illustrates the time-varying salinity data in the Southern California Bight region generated by the Regional Ocean Modeling

[†]This work is an extension of a proceeding paper [Ma et al., 2017].

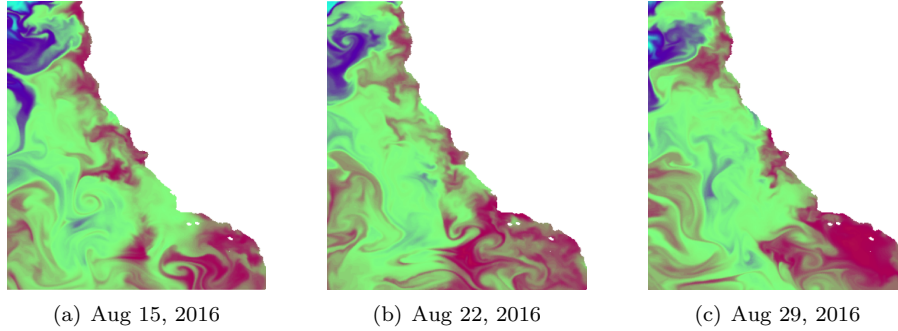


Figure 1: Ocean salinity data in the Southern California Bight region generated by the Regional Ocean Modeling System (ROMS) [Shchepetkin and McWilliams, 2005]. Color indicates levels of salinity content.

System (ROMS) [Shchepetkin and McWilliams, 2005]. The variations in spatial and temporal dimensions entail that the collected data must contain the maximal information in order to provide a good estimate of the environment at any time [Meliou et al., 2007, Ouyang et al., 2014].

Increasingly, a variety of autonomous robotic systems, including marine vehicles [Fiorelli et al., 2003], aerial vehicles [Watts et al., 2012], and ground vehicles [Trincavelli et al., 2008], are designed and deployed for environmental monitoring in order to replace conventional methods of deploying static sensors to areas of interest [Oliveira and Rodrigues, 2011]. Particularly, autonomous underwater vehicles (AUVs) such as marine gliders are becoming popular due to their long-range (hundreds of kilometers) and long-duration (weeks even months) monitoring capabilities [Miles et al., 2015, Paley et al., 2008, Leonard et al., 2010].

We are interested in estimating the current state of the environment and providing a nowcast (instead of forecast) of the environment, by navigating robots to collect the information used for such estimation. To model spatial phenomena, a common approach is to use a rich class of techniques called Gaussian Processes (GPs) [Rasmussen and Williams, 2005, Singh et al., 2007, Ouyang et al., 2014] in spatial statistics. In this work, we also employ this broadly-adopted approach to build and learn an underlying environmental model that describes the “data map” of a target attribute of interest.

Still, there are a few challenges to address:

- The first challenge lies in the acquirement of the most useful sensing inputs for learning the model, i.e., since samples from different locations are not equally important, thus we wish to seek for the samples that best describe the environment. Navigating robots to obtain such samples is called *informative planning* [Singh et al., 2007, Binney et al., 2013]. In this work, we utilize the mutual information between visited locations and the remainder of the space to characterize the amount of information (information gain) collected.
- The second challenge is the relaxation of the prohibitive computational cost for maintaining the model. The most accurate way to estimate a latent model is to use all historical sensing data. However, since the environmental monitoring task can be long-range and long-term, the data size continuously grows and it will eventually “explode”. Consequently, an efficient estimator will need to dynamically select only the most informative data while abandoning the samples that contribute less.

We propose an informative planning and online learning approach for long-term environmental monitoring. Our objective is to construct and maintain an environmental model by navigating a robot to collect data with the greatest information. We have developed a framework that includes two important components. The first component provides a high level path planning solutions where the path/navigation waypoints

represent those most informative sampling locations. The second component is a learning module where the hyperparameters of the Gaussian Process are learned via a data-driven Bayesian inference method. To achieve online computation and planning, we employ a sparse variant of GP so that only a fixed size of those most important samples are dynamically retained, and the updated environmental model with new incoming data is in turn used to improve the high level planning component.

2 Related Work

Environmental sensing and monitoring allow scientists to assess the processes of a particular environment, and have been used in a broad range of applications. For example, an array of sensor networks can be deployed to detect forest fires [Lloret et al., 2009] and volcano activities [Werner-Allen et al., 2006]; unmanned aerial vehicles (UAVs) have been used to estimate yields of crops or fruits [Nuske et al., 2011, Yang, 2012] and to study spatial ecology and its spatiotemporal dynamics [Anderson and Gaston, 2013]; with a capacity of performing long-range and long-term tasks, marine robots can collect large-area ocean data [Smith et al., 2011] and trace chemical plumes [Farrell et al., 2005, Hajieghrary et al., 2015]; autonomous boats have been used to monitor fish schools [Tokekar et al., 2010].

Planning and environment monitoring are two big and well studied topics. Here we focus on reviewing the works that are related to the informative planning and the model prediction with GPs. Formally, the planning method that navigates robots to collect and maximize information gain through sampling the environment is called *informative planning* [Singh et al., 2007, Binney et al., 2013]. A variety of methodologies have been proposed to tackle the informative planning problem, among which the most investigated approaches belong to the *nonmyopic* framework. Formally, the term *myopic* means that the path waypoints are computed independently and greedily, without considering the cost and consequences of making observations in a long horizon of future. Instead, a nonmyopic strategy performs optimization and computes a series of waypoints by considering the effect of later time-steps [Meliou et al., 2007].

Representative nonmyopic informative planning approaches include, for example, a *recursive-greedy* based algorithm [Singh et al., 2007] where the informativeness is generalized as a *submodular set function* built on which a sequential-allocation mechanism is designed in order to obtain subsequent waypoints. This recursive-greedy framework has been extended by taking into account the avoidance of shipping lanes [Binney et al., 2010] and diminishing returns [Binney et al., 2013]. Differing from above mechanisms where the path waypoints are built by separate searching techniques with the informativeness as some utility function, Low [Low, 2009] proposed a differential entropy based planning method in which a batch of waypoints can be obtained through solving a dynamic program. Such a framework has been extended to approaches incorporating mutual information [Cao et al., 2013] and Markovian [Low et al., 2011] optimization criteria. However, these approaches are formulated with an assumption that the underlying map is in a regular shape (e.g., a parallelogram shape) and the map is transected (sliced) column-wise, so that each algorithmic iteration computes waypoints within a column and the navigation paths are obtained by connecting those waypoints among the pairwise adjacent columns. We recently proposed an informative planning method based on the dynamic programming structure in order to compute the informative waypoints across an arbitrary continuous space [Ma et al., 2016a]. This nonmyopic method has also been combined with Markov Decision Process to cope with robot’s action uncertainty caused by external disturbances. In addition, there are also many methods optimizing over complex planning and information constraints (e.g., [Soltero et al., 2012, Yu et al., 2014]).

To model spatial phenomena in a continuous domain, a broadly adopted method is the Gaussian Process which is a generic supervised learning method designed to solve regression and probabilistic classification problems [Rasmussen and Williams, 2005, Singh et al., 2007, Ouyang et al., 2014]. The regression property of GP has manifested itself as a powerful tool for predicting environment states based on which the environment can be monitored. For instance, GP has been used to design placement patterns of static sensors in a sensor network so that the environment model can be predicted with a solution that is near-

optimal [Guestrin et al., 2005]. Optimization of static sensor placement has been applied for modeling indoor 3-dimensional environment [Erickson et al., 2015] and outdoor urban environment [Liu et al., 2016] through appropriate kernels (covariance functions). In addition to the applications in sensor networks, GPs are also being exploited in robotics research. For example, GP has been used to provide a measure of uncertainty to guide sensor-centric robot localization [Brooks et al., 2008]; through combining with mutual information and Bayesian optimization, GP has been utilized to guide robots to explore unknown static environment [Bai et al., 2016]. In dynamic environment settings, variants of GPs have also been employed to learn uncertainty models of ocean processes to assist in the operation of AUVs in the ocean [Hollinger et al., 2013, Hollinger et al., 2016]. By integrating with a deterministic vehicle routing scheme, we also proposed a method for informative ocean sampling and monitoring in dynamic ocean environments with multi-robot systems [Ma et al., 2016b].

This presented work is also built on top of a GP which is used to describe an interested environmental attribute (e.g., we measure the salinity and build a continuous salinity data-map). Our GP model here is coupled with the information-theoretic framework where we compute the mutual information between visited locations and the remainder of the space to characterize the amount of information collected, through which we use the informative planning framework to compute high-level informative navigation waypoints. Different from methods mentioned above, in this work we consider dynamic environments that vary both spatially and temporally. This means that the latent parameters that support/characterize the GP model need to be updated too in order to accurately reflect the ground-truth environment in a timely manner.

To improve GP’s prediction accuracy, the choice over different prior covariance functions and the update of its hyperparameters are crucial, especially in the scenarios involving many spatiotemporal dynamics. This problem is typically referred as *Model Selection and Adaptation of Hyperparameters* [Rasmussen and Williams, 2005]. Particularly, the adaptation of hyperparameters can be updated using a data-driven approach. The most common ones are done by maximizing the marginal likelihood or minimizing the generalization error using cross-validation approach. For the case of GP classification, other optimization criteria such as *Alignment* [Cristianini et al., 2001] can also be adopted.

A critical problem for persistent (long-term even life-long) tasks that one must consider is the large-scale data accumulated. Although abundant data might predict the most accurate model, in practice large amounts of data can exceed the robot’s onboard computational capacity. Methods for reducing the computing burdens of GPs have been previously investigated. For example, GP regressions can be done in a real-time fashion where the problem can be estimated locally with local data [Nguyen-tuong and Peters, 2008]. Another representative framework is a sparse representations of the GP model [Csato and Opper, 2002] which is based on a combination of a Bayesian online algorithm together with a sequential construction of the most relevant subset of the data. This method allows the model to be refined in a recursive way as the data streams in. The framework has been further extended to many application domains such as visual tracking [Ranganathan et al., 2011].

Recently, we proposed an informative planning and online learning approach for long-term spatiotemporal environmental monitoring [Ma et al., 2017] with the objective to construct and maintain a spatiotemporal environmental model by navigating the robot to the most informative regions to collect data with the greatest information. The method has integrated a sparse variant of GPs so that both the model and hyperparameters can be improved online with dynamic but a fixed size of data. Then the ameliorated environment model is in turn used to improve the planning component at appropriate re-planning moments. This paper is an extension version of the the previous work [Ma et al., 2017] and distinguishes itself from earlier work in the following aspects. First, we have extended the work by adding many necessary details so that the method is theoretically complete and in-depth. Second, we have added new simulation results to capture data (environmental attribute) variations in the temporal dimension. Third, we have validated the approach in field trials, where we implemented our method on an ASV developed in our lab, and have deployed it in a lake to collect data and estimate a continuous data map of the lake.

3 Preliminaries

In this section, we briefly present the preliminary background for the GP-based environmental modeling.

3.1 Gaussian Process Regression on Spatial Data

A Gaussian Process is defined as a collection of random variables where any finite number of which have a joint Gaussian distribution. GP's prediction behavior is determined by the prior covariance function (also known as *kernel*) and the training points. The prior covariance function describes the relation between two independent data points and it typically comes with some free hyperparameters to control the relation.

Formally, let X be the set of n training points associated with target values, \mathbf{y} , and let X_* be the testing points. The predictive equations of the GP regression can be summarized as:

$$\begin{aligned} \mathbf{f}_*|X, \mathbf{y}, X_* &\sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*)) \\ \bar{\mathbf{f}}_* &\triangleq \mathbb{E}[\mathbf{f}_*|X, \mathbf{y}, X_*] = K(X_*, X)K(X, X)^{-1}\mathbf{y} \\ \text{cov}(\mathbf{f}_*) &= K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*) \end{aligned} \quad (1)$$

where $K(\cdot, \cdot)$ denotes a matrix where each entry is the covariance evaluated by the kernel function. For example, $K(X, X_*)$ is a $|X| \times |X_*|$ matrix evaluated by a pre-selected kernel function for all pairwise data points in X and X_* . A widely adopted choice of kernel function for spatial data is the *squared exponential automatic relevance determination* function [Neal, 1996]:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T M (\mathbf{x} - \mathbf{x}')\right) + \sigma_n^2 \delta_{\mathbf{x}\mathbf{x}'} \quad (2)$$

where $M = \text{diag}(\mathbf{l})^{-2}$. The parameters \mathbf{l} are the *length-scales* in each dimension of \mathbf{x} and determine the level of correlation (each l_i models the degree of smoothness in the spatial variation of the measurements in the i th dimension of the feature vector \mathbf{x}). σ_f^2 and σ_n^2 denote the variances of the signal and noise, respectively. $\delta_{\mathbf{x}\mathbf{x}'}$ is the Kronecker delta function which is 1 if $\mathbf{x} = \mathbf{x}'$ and zero otherwise.

3.2 Estimation of Hyperparameters Using Training Data

Let $\boldsymbol{\theta} \triangleq \{\sigma_n^2, \sigma_f^2, \mathbf{l}\}$ be the set of hyperparameters in the kernel function. We are interested in estimating these hyperparameters so that the kernel function can describe the underlying phenomena as accurate as possible. A common approach to learning the set of hyperparameters is via maximum likelihood estimation combined with k -fold cross-validation (CV) [Rasmussen and Williams, 2005]. An extreme case of the k -fold cross-validation is when k is equal to the number of training points n . It is also known as the *leave-one-out cross-validation* (LOO-CV). Mathematically, the log-likelihood for the leaving out training case i is

$$\log p(y_i|X, \mathbf{y}_{-i}, \boldsymbol{\theta}) = -\frac{1}{2} \log[\text{var}(\bar{f}_i)] - \frac{(y_i - \bar{f}_i)^2}{2\text{var}(\bar{f}_i)} - \frac{1}{2} \log(2\pi) \quad (3)$$

where \mathbf{y}_{-i} denotes all targets in the training set except the one with index i , and \bar{f}_i and $\text{var}(\bar{f}_i)$ are calculated according to Eq. (1). The log-likelihood of LOO is therefore

$$L_{LOO}(X, \mathbf{y}, \boldsymbol{\theta}) = \sum_{i=1}^n \log p(y_i|X, \mathbf{y}_{-i}, \boldsymbol{\theta}). \quad (4)$$

Notice that in each of the $|\mathbf{y}|$ LOO-CV iterations, the inverse of the training set covariance matrix, $K(X_{-i}, X_{-i})^{-1}$, is needed, which is costly if computed repeatedly. This can actually be computed efficiently from the inverse of the complete covariance matrix using *inversion by partitioning* [Press et al., 1996].

The resulting predictive mean and variance can then be formulated as

$$\begin{aligned}\bar{f}_i &= y_i - [K(X, X)^{-1} \mathbf{y}]_i / [K(X, X)^{-1}]_{ii} \\ \text{var}(\bar{f}_i) &= 1 / [K(X, X)^{-1}]_{ii}\end{aligned}\tag{5}$$

To obtain the optimal values of hyperparameters $\boldsymbol{\theta}$, we can compute the partial derivatives of L_{LOO} and use the gradient optimization techniques. The partial derivatives of L_{LOO} using chain rules is

$$\frac{\partial L_{LOO}}{\partial \theta_j} = \sum_{i=1}^n \left(\frac{\partial \log p(y_i | X, \mathbf{y}_{-i}, \boldsymbol{\theta})}{\partial \bar{f}_i} \frac{\partial \bar{f}_i}{\partial \theta_j} + \frac{\partial \log p(y_i | X, \mathbf{y}_{-i}, \boldsymbol{\theta})}{\partial \text{var}(\bar{f}_i)} \frac{\partial \text{var}(\bar{f}_i)}{\partial \theta_j} \right)\tag{6}$$

To calculate the partial derivatives of L_{LOO} , we need the partial derivatives of the LOO-CV predictive mean and variances from Eq. (5).

$$\begin{aligned}\frac{\partial \bar{f}_i}{\partial \theta_j} &= \frac{[Z_j \boldsymbol{\alpha}]_i}{[K(X, X)^{-1}]_{ii}} - \frac{\boldsymbol{\alpha}_i [Z_j K(X, X)^{-1}]_{ii}}{[K(X, X)^{-1}]_{ii}^2} \\ \frac{\partial \text{var}(\bar{f}_i)}{\partial \theta_j} &= \frac{[Z_j K(X, X)^{-1}]_{ii}}{[K(X, X)^{-1}]_{ii}^2}\end{aligned}\tag{7}$$

where $\boldsymbol{\alpha} = K(X, X)^{-1} \mathbf{y}$ and $Z_j = K(X, X)^{-1} \frac{\partial K(X, X)}{\partial \theta_j}$.

Substitute $\frac{\partial \bar{f}_i}{\partial \theta_j}$ and $\frac{\partial \text{var}(\bar{f}_i)}{\partial \theta_j}$ in Eq. (6) and calculate the partial derivative from Eq. (3), we have

$$\frac{\partial L_{LOO}}{\partial \theta_j} = \sum_{i=1}^n \frac{1}{[K(X, X)^{-1}]_{ii}} \left(\alpha_i [Z_j \boldsymbol{\alpha}]_i - \frac{1}{2} \left(1 + \frac{\alpha_i^2}{[K(X, X)^{-1}]_{ii}} \right) [Z_j K(X, X)^{-1}]_{ii} \right),\tag{8}$$

Using the standard gradient method, we update each θ_j iteratively:

$$\theta_j^{(t+1)} = \theta_j^{(t)} + \eta \frac{\partial L_{LOO}}{\partial \theta_j^{(t)}},\tag{9}$$

where η is the learning rate.

4 Online Learning and Informative Planning

As aforementioned, one limitation of GPs for a long-term mission is the memory requirement for large (possibly infinite) training sets. In our system, we borrow the idea of Sparse Online Gaussian Process (SOGP) [Csat  and Oppel, 2002] to overcome this limitation. The method is based on a combination of a Bayesian online algorithm together with a sequential subsampling of the data which best describes a latent model.

4.1 Bayesian Learning with Gaussian Processes

Given a prior GP $\hat{p}_t(\mathbf{f})$ at time t , when a new data point $(\mathbf{x}_{t+1}, y_{t+1})$ at time $t+1$ comes in, it is incorporated by performing a Bayesian update to yield a posterior.

$$p_{\text{post}}(\mathbf{f}) = \frac{p(y_{t+1} | \mathbf{f}) \hat{p}_t(\mathbf{f})}{\mathbb{E}_{\hat{p}_t(\mathbf{f})}[p(y_{t+1} | \mathbf{f}_D)]},\tag{10}$$

where $\mathbf{f} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_M)]^T$ denotes a set of function values, and $\mathbf{f}_D \subseteq \mathbf{f}$ where \mathbf{f}_D is the set of $f(\mathbf{x}_i) = f_i$ with \mathbf{x}_i in the training set. In general, $p_{\text{post}}(\mathbf{f})$ is no longer Gaussian unless the likelihood

itself is also Gaussian. Therefore, $p_{post}(\mathbf{f})$ is projected onto the closest GP, \hat{p}_{t+1} where $\hat{p}_{t+1} = \arg \min_{\hat{p}} \text{KL}(p_{post}(\mathbf{f}) || \hat{p})$. (KL is the Kullback-Leibler divergence that is used to measure the difference between two probability distributions.) It is shown in [Oppel, 1998] that the projection results in a good matching of the first two moments (mean and covariance) of p_{post} and the new Gaussian posterior \hat{p}_{t+1} . By following the lemma of [Csat and Oppel, 2002], we arrive at the parametrization for the approximate posterior GP at time $(t + 1)$ as a function of the kernel and likelihoods.

$$\begin{aligned}\bar{f}_* &= \sum_{i=1}^{t+1} k(\mathbf{x}_*, \mathbf{x}_i) \alpha_{t+1}(i) = \boldsymbol{\alpha}_{t+1}^T \mathbf{k}_{\mathbf{x}_*, t+1} \\ \text{var}(f_*) &= k(\mathbf{x}_*, \mathbf{x}_*) + \sum_{i,j=1}^{t+1} k(\mathbf{x}_*, \mathbf{x}_i) [C_{t+1}]_{ij} k(\mathbf{x}_j, \mathbf{x}_*) \\ &\triangleq k(\mathbf{x}_*, \mathbf{x}_*) + \mathbf{k}_{\mathbf{x}_*, t+1}^T C_{t+1} \mathbf{k}_{\mathbf{x}_*, t+1}\end{aligned}\tag{11}$$

where $\mathbf{k}_{\mathbf{x}_*, t+1} = [k(\mathbf{x}_1, \mathbf{x}_*), \dots, k(\mathbf{x}_{t+1}, \mathbf{x}_*)]^T$, and $\boldsymbol{\alpha}_{t+1}$ and C_{t+1} are updated using

$$\begin{aligned}\boldsymbol{\alpha}_{t+1} &= T_{t+1}(\boldsymbol{\alpha}_t) + q_{t+1} \mathbf{s}_{t+1} \\ C_{t+1} &= U_{t+1}(C_t) + r_{t+1} \mathbf{s}_{t+1} \mathbf{s}_{t+1}^T \\ \mathbf{s}_{t+1} &= T_{t+1}(C_t \mathbf{k}_{\mathbf{x}_*, t+1}) + \mathbf{e}_{t+1} \\ q_{t+1} &= \frac{\partial}{\partial \mathbb{E}_{\hat{p}_t(\mathbf{f})}[\mathbf{f}_{t+1}]} \log \mathbb{E}_{\hat{p}_t(\mathbf{f})}[p(y_{t+1} | \mathbf{f}_{t+1})] \\ r_{t+1} &= \frac{\partial^2}{\partial \mathbb{E}_{\hat{p}_t(\mathbf{f})}[\mathbf{f}_{t+1}]^2} \log \mathbb{E}_{\hat{p}_t(\mathbf{f})}[p(y_{t+1} | \mathbf{f}_{t+1})]\end{aligned}\tag{12}$$

where \mathbf{e}_{t+1} is the $(t + 1)$ -th unit vector. The operator T_{t+1} (U_{t+1}) is defined to extend a t -dimensional vector (matrix) to a $(t + 1)$ -dimensional one by appending zero at the end of the vector (zeros at the last row and column of the matrix). Initially at time 0, $\boldsymbol{\alpha}_0$ and C_0 are set to be zero-sized vector and matrix. For the regression with Gaussian noise (variance σ_0^2), The expected likelihood is a normal distribution with mean \bar{f}_* and variance $\text{var}(f_*) + \sigma_0^2$. Hence, the logarithm of the expected likelihood is:

$$\log \mathbb{E}_{\hat{p}_{t+1}(\mathbf{f})}[p(y_t | \mathbf{f}_t)] = -\frac{1}{2} \log[2\pi(\text{var}(f_*) + \sigma_0^2)] - \frac{(y_t - \bar{f}_*)^2}{2(\text{var}(f_*) + \sigma_0^2)},\tag{13}$$

and the first and second derivatives with respect to the mean \bar{f}_* are

$$\begin{aligned}q_t &= \frac{y_t - \bar{f}_*}{\text{var}(f_*) + \sigma_0^2}, \\ r_t &= -\frac{1}{\text{var}(f_*) + \sigma_0^2},\end{aligned}\tag{14}$$

where both q_t and r_t are scalars.

4.2 Online Process of Sparse Samples

To prevent the unbounded growth of memory requirement due to the increase of data, it is necessary to limit the number of the training points which are stored in a *basis vector set* (*BV-set*), while preserving the predictive accuracy of the model. This is done in two different stages.

First, when a new training point $(\mathbf{x}_{t+1}, y_{t+1})$ at time $t + 1$ arrives, if there exists a $\hat{\mathbf{e}}_t$ such that the relation

$$k(\mathbf{x}, \mathbf{x}_{t+1}) = \sum_{i=1}^t \hat{\mathbf{e}}_{t+1}(i) k(\mathbf{x}, \mathbf{x}_i)\tag{15}$$

holds for all \mathbf{x} in the input space, it essentially means the feature vector $\Phi(\mathbf{x}_{t+1})$ lies exactly on the space spanned by the current BV-set, $(\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_t))$, where Φ transforms input \mathbf{x} into feature space. If this is the case, then the GP can still be modeled using only the first t inputs, but with “re-normalized” parameters $\hat{\alpha}_{t+1}$ and \hat{C}_{t+1} , which is done by updating $\hat{\mathbf{s}}_{t+1}$ via:

$$\hat{\mathbf{s}}_{t+1} = C_t \mathbf{k}_{\mathbf{x}_{t+1}, t} + \hat{\mathbf{e}}_{t+1}. \quad (16)$$

However, the exact $\hat{\mathbf{e}}_{t+1}$ does not always exist for most kernels and inputs \mathbf{x}_{t+1} . Nevertheless, we could try to approximate it by minimizing the error measure

$$\|k(\mathbf{x}, \mathbf{x}_{t+1}) - \sum_{i=1}^t \hat{\mathbf{e}}_{t+1}(i) k(\mathbf{x}, \mathbf{x}_i)\|^2, \quad (17)$$

where $\|\cdot\|$ is the norm operation. The minimization ([Csat and Oppner, 2002]) of Eq. (17) leads to

$$\hat{\mathbf{e}}_{t+1} = Q_t^{-1} \mathbf{k}_{\mathbf{x}_{t+1}, t}, \quad (18)$$

where $Q_t = K(X_t, X_t)^{-1}$ is the inversion of the full kernel matrix. Note that the re-normalization update in Eq. (16) will be done only if the approximation error does not exceed some predefined threshold ω , otherwise, the sample is added into the BV-set as described in 4.1. Let the quantity of the approximation error be γ_{t+1} . Specifically, it is the squared norm of the “residual vector” from the projection in the feature space spanned by the current BV-set.

$$\gamma_{t+1} = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}_{\mathbf{x}_{t+1}, t}^T Q_t \mathbf{k}_{\mathbf{x}_{t+1}, t}, \quad (19)$$

Essentially, γ_{t+1} can also be thought of as a form of “novelty” for the new training point $(\mathbf{x}_{t+1}, y_{t+1})$.

Second, when the size of BV-set exceeds the memory limit (or any pre-defined limit), m , a score measure is used to pick out the lowest one and remove it from the existing BV-set. Formally, let ϵ_i be the scoring function for the i th element in the BV-set,

$$\epsilon_i = \frac{[\alpha_{t+1}]_i}{[Q_{t+1}]_{ii}} \quad (20)$$

which is a measure of change on the expected posterior mean of a sample due to sparse approximation. Assume the j th element in BV-set is the one with the lowest ϵ , the removal of it requires a re-update of parameters α_{t+1} , C_{t+1} and Q_{t+1}

$$\begin{aligned} \hat{\alpha}_{t+1} &= \alpha^{(t)} - \alpha^j \frac{Q^j}{q^j} \\ \hat{C}_{t+1} &= C^{(t)} + c^j \frac{Q^j Q^{jT}}{q^{j2}} - \frac{1}{q^j} [Q^j C^{jT} + C^j Q^{jT}] \\ \hat{Q}_{t+1} &= Q^{(t)} - \frac{Q^j Q^{jT}}{q^j}, \end{aligned} \quad (21)$$

where $C^{(t)}$ is the resized matrix by removing the j th column and the j th row from C_{t+1} , C^j is the j th column of C_{t+1} excluding the j th element and $c^j = [C_{t+1}]_{jj}$. Similar operations apply for $Q^{(t)}$, Q^j , q^j , $\alpha^{(t)}$, and α^j .

4.3 Environment Representation and Informative Sampling

To facilitate the computation of future informative sampling locations, we discretize the environment into a grid map where each grid represents a possible sampling location. The mean and variance of the measurement at each grid can be predicted via the SOGP model. We use the *mutual information* between the visited locations and the remainder of the space to characterize the amount of information (information gain)

collected. Formally, the mutual information between two sets of sampling locations, A, B can be evaluated as:

$$I(Z_A; Z_B) = I(Z_B; Z_A) = H(Z_A) - H(Z_A|Z_B). \quad (22)$$

The entropy $H(Z_A)$ and conditional entropy $H(Z_A|Z_B)$ can be calculated by

$$\begin{aligned} H(Z_A) &= - \int p(Z_A) \log p(Z_A) d(Z_A) = \frac{1}{2} \log \left((2\pi e)^k |\Sigma_{AA}| \right) \\ H(Z_A|Z_B) &= \frac{1}{2} \log \left((2\pi e)^k |\Sigma_{A|B}| \right) \end{aligned} \quad (23)$$

where k is the size of A . The covariance matrix Σ_{AA} and $\Sigma_{A|B}$ can essentially be calculated from the posterior GP described in Eq. (11).

To compute the future sampling locations, let X denote the entire sampling space (all grids), and Z_X be measurements for data points in X . The objective is to find a subset of sampling points, $P \subset X$ with a size $|P| = n$, which gives us the most information for predicting our model. This is equivalent to the problem of finding new sampling points in the un-sampled space that maximizes the mutual information between sampled locations and un-sampled part of the map. The optimal subset of sampling points, P^* , with maximal mutual information is

$$P^* = \arg \max_{P \in \mathcal{X}} I(Z_P; Z_{X \setminus P}) \quad (24)$$

where \mathcal{X} represents all possible combinatorial sets, each of which is of size n . P^* can be computed efficiently using a dynamic programming (DP) scheme [Ma et al., 2016a]. Here is the idea: Let $\mathbf{x}_i \in X$ denote an arbitrary sampling point at DP stage i and $\mathbf{x}_{a:b}$ represent a sequence of sampling points from stage a to stage b . The mutual information between the desired sampling points (which eventually form P) and the remaining map can then be written as $I(Z_{\mathbf{x}_{1:n}}; Z_{X \setminus \{\mathbf{x}_{1:n}\}})$, which can be expanded using the chain rule:

$$I(Z_{\mathbf{x}_{1:n}}; Z_{X \setminus \{\mathbf{x}_{1:n}\}}) = I(Z_{\mathbf{x}_1}; Z_{X \setminus \{\mathbf{x}_{1:n}\}}) + \sum_{i=2}^n I(Z_{\mathbf{x}_i}; Z_{X \setminus \{\mathbf{x}_{1:n}\}} | Z_{\mathbf{x}_{1:i-1}}). \quad (25)$$

One can utilize this form of mutual information to calculate \mathbf{x}_i step by step, however, at every stage i before the final stage, the entire unobserved set $X \setminus \{\mathbf{x}_{1:n}\}$ is not known in advance, therefore we make an approximation

$$I(Z_{\mathbf{x}_{1:n}}; Z_{X \setminus \{\mathbf{x}_{1:n}\}}) \approx I(Z_{\mathbf{x}_1}; Z_{X \setminus \{\mathbf{x}_1\}}) + \sum_{i=2}^n I(Z_{\mathbf{x}_i}; Z_{X \setminus \{\mathbf{x}_1, \dots, \mathbf{x}_i\}} | Z_{\mathbf{x}_{1:i-1}}), \quad (26)$$

Eq. (26) can now be expressed in a recursive form, i.e. for stages $i = 2, \dots, n$, the value $V_i(\mathbf{x}_i)$ of \mathbf{x}_i is:

$$V_i(\mathbf{x}_i) = \max_{\mathbf{x}_i \in X \setminus \{\mathbf{x}_1, \dots, \mathbf{x}_{i-1}\}} I(Z_{\mathbf{x}_i}; Z_{X \setminus \{\mathbf{x}_1, \dots, \mathbf{x}_i\}} | Z_{\mathbf{x}_{1:i-1}}) + V_{i-1}(\mathbf{x}_{i-1}),$$

with a recursion base case $V_1(\mathbf{x}_1) = I(Z_{\mathbf{x}_1}; Z_{X \setminus \{\mathbf{x}_1\}})$. Then with the optimal solution in the last stage, $\mathbf{x}_n^* = \arg \max_{\mathbf{x}_n \in X} V_n(\mathbf{x}_n)$, we can backtrack all optimal sampling points until the first stage \mathbf{x}_1^* , and obtain $P^* = \{\mathbf{x}_1^*, \mathbf{x}_2^*, \dots, \mathbf{x}_n^*\}$. The whole computational process for the information-driven planner is pseudo-coded in Alg. 1.

Note that, the informativeness maximization procedure only outputs batches of sampling points, but does not convey any information of “a path” which is a sequence of ordered waypoints. Therefore, these sampling points are post-processed with a customized Travelling Salesman Problem (TSP) [Laporte, 1992] solver to generate a shortest path but without returning to the starting point (by setting all edges that return to the starting point with 0 cost). We then route the robot along the path from its initial location to visit the remaining path waypoints.

Algorithm 1: Information-Driven Planner

```
1 Given the desired number of waypoints  $n$ 
2 foreach  $\mathbf{x} \in X$  do
3    $V_1(\mathbf{x}) = I(Z_{\mathbf{x}}; Z_{X \setminus \{\mathbf{x}\}})$ 
4 foreach  $i = 2$  to  $n$  do
5   foreach  $\mathbf{x} \in X$  do
6     initialize  $V_i(\mathbf{x}) = -\infty$ 
7   foreach  $\mathbf{x}_{i-1} \in X$  do
8     foreach  $\mathbf{x}_i \in X \setminus \{\mathbf{x}_{1:i-1}\}$  do
9        $V_i(\mathbf{x}_i) = \max(I(Z_{\mathbf{x}_i}; Z_{X \setminus \{\mathbf{x}_{1:i}\}} | Z_{\mathbf{x}_{1:i-1}}) + V_{i-1}(\mathbf{x}_{i-1}), V_i(\mathbf{x}_i))$ 
10  $\mathbf{x}_n^* = \arg \max_{\mathbf{x}_n \in X} V_n(\mathbf{x}_n)$ 
11 Backtrace to get  $\mathbf{x}^* \triangleq (\mathbf{x}_1^*, \mathbf{x}_2^*, \dots, \mathbf{x}_n^*)$ 
```

4.4 Overall Framework

For a dynamic environment, the prediction accuracy of GP degrades as time elapses because it does not incorporate the temporal variation of the environment. To address this issue, we re-estimate the hyperparameters at appropriate moments. The re-estimation triggering mechanism depends on two factors:

- The first factor stems from the computational concern. Since any re-estimation will be immediately followed by a re-planning of the future routing path, and because the computation time for the path planning is much more costly than that of the hyperparameter re-estimation. Thus, an appropriate frequency for the simultaneous re-estimation and re-planning needs to be determined to match the computational constraint.
- The second factor relates to the intensity of spatiotemporal variations. Since the kernel function that describes two points' spatial relation is an indicator of a GP's prediction capacity, thus the repetitive hyperparameter re-estimation of the kernel function should reflect the variation intensity of the environment.

In our implementation, we use a measure, $\rho \in [0, 1]$, to decide the moment for triggering the re-estimation and re-planning processes. The measure ρ represents the proportion of samples that are recently added to the current BV-set since last re-estimation. The hyperparameter re-estimation and path re-planning are carried out if ρ is above certain pre-defined threshold, ρ_0 . Roughly, ρ_0 can be defined to be inversely proportional to the computational power and the intensity of environmental variation, and the higher the threshold, the less frequent the re-estimation. The whole informative planning and online learning framework is pseudo-coded in Alg. 2.

5 Simulation with Ocean Data

The proposed method is a generic framework which can be applied to any kinds of environments where the quantity of the interested phenomena varies spatially and temporally. That is, the environment can be terrestrial, aquatic or aerial and the dimension of the environment can be arbitrary too. Such examples include the concentration of pollutants in the air or the density of plankton in the ocean. Here, we validated our method in the scenario of ocean monitoring. The simulation environment was constructed as a two dimensional ocean surface and we tessellated the environment into a grid map. In our experiments, we use ocean salinity data recently observed and post-processed by Regional Ocean Modeling System (ROMS) [Shchepetkin and McWilliams, 2005]. Fig. 2(a) shows the post-processed salinity data in the Southern California

Algorithm 2: Online Learning and Informative Planning

```
1 Initialize SOGP
2 while true do
3    $\rho = 0$  /* for hyperparameter re-estimation */
4   Calculate sampling locations as described in 4.3
5   Compute informative routing path,  $P$ , based on the sampling locations generated
6   foreach point  $p \in P$  do
7     Do sampling on  $p$  to get a scalar value  $v$ 
8     Use  $(p, v)$  as a training point to update SOGP described in 4.1 and 4.2
9     if  $(p, v)$  replaces some sample in the BV-set then
10      Increase  $\rho$ 
11     if  $\rho > \rho_0$  then
12      Do hyperparameter re-estimation described in 3.2
13      break
```

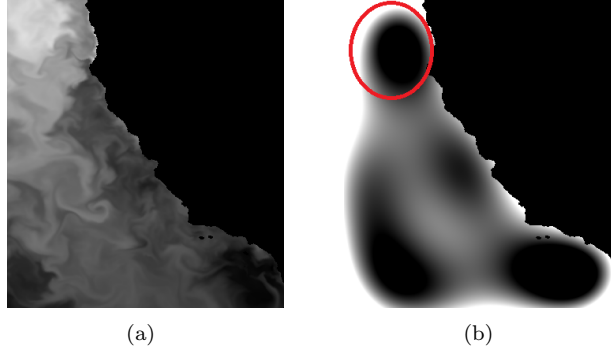


Figure 2: (a) Salinity data obtained from ROMS. It is treated as a ground truth throughout the paper; (b) The predicted model using GP without data-driven hyperparameter optimization.

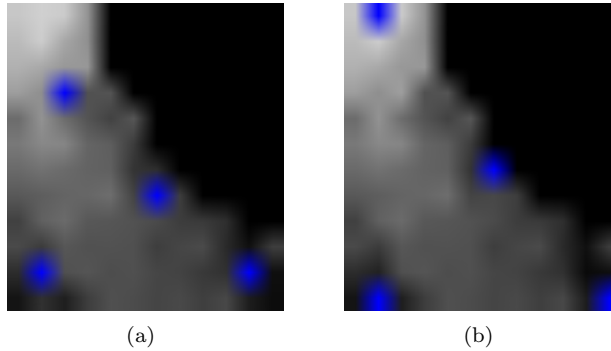


Figure 3: Informative sampling locations before they are post-processed as paths. (a) Results using hyperparameters empirically set: $\{\sigma_n^2 = \exp(-2), \sigma_f^2 = \exp(2), l_x = \exp(1), l_y = \exp(1)\}$; (b) Results using hyperparameters learned from data collected: $\{\sigma_n^2 = \exp(-4.6), \sigma_f^2 = \exp(6.8), l_x = \exp(3.4), l_y = \exp(3.2)\}$.

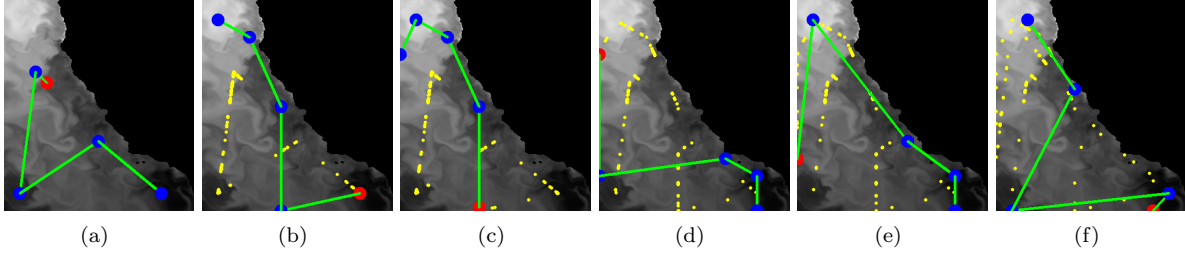


Figure 4: (a)-(f) Informative paths resulted from subsequent re-plannings. The red and blue points represent the robot’s starting locations and the informative sampling locations, respectively. The latitude of the ocean area is ranged from 31.3 to 43, and the longitude is ranged from 232.5 to 243. The robot initially launched at (79, 236). The yellow dots denote the points stored in the SOGP BV-set.

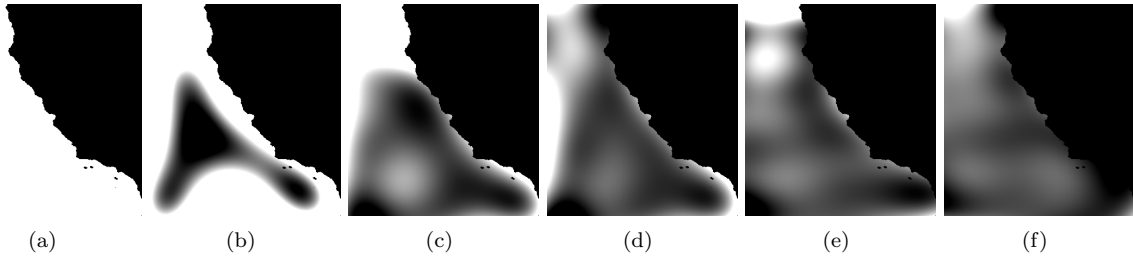


Figure 5: (a)-(f) The learned environment models. Each corresponds to a step in Fig. 4.

Bight region. The data is represented as a scalar field (the black regions represent lands while the gray areas denote ocean), which is used as the ground truth for comparison.

We implemented a sparse online variant of GP (SOGP) built upon the open-source library *libgp* [Blum and Riedmiller, 2013]. A careful down-sampling of ROMS data to a desired resolution is performed to alleviate the computational cost for generating informative sampling locations. The resolution of the grid map is 351×391 , whereas the resolution for the sampling points generation (path planning) is 12×12 .

First, we investigate the model prediction accuracy using un-tuned hyperparameters, i.e., hyperparameter values are set empirically/manually instead of data-driven. Fig. 2(b) shows a prediction result with 50 prior random samples and manually set hyperparameters $\theta_0 = \{\sigma_n^2 = \exp(-2), \sigma_f^2 = \exp(2), l_x = \exp(1), l_y = \exp(1)\}$. We can observe that the prediction does not match well with the ground truth (see the area circled in red). This is because that, as mentioned in 3.1 the hyperparameters of a GP “control” the spatial relation between two independent measurements (data points in the field), and a particular set of samples need a particular configuration of hyperparameters for the best regression. Thus, empirically setting values is very likely to be suboptimal. In addition to that, we have also investigated and compared the generated informative sampling points (locations) using empirical and data-driven hyperparameters. Here the data-driven method produces optimized hyperparameters $\hat{\theta} = \{\sigma_n^2 = \exp(-4.6), \sigma_f^2 = \exp(6.8), l_x = \exp(3.4), l_y = \exp(3.2)\}$, from which we can observe that every single parameter of the optimized set has changed in order to fit the data. Fig. 3(a) and 3(b) show generated informative sampling points from the environmental model with manually-set and data-driven hyperparameters, respectively. We can see that the relative distances between points (and the covered areas) in Fig. 3(b) are larger than those in Fig. 3(a). This is mainly affected by \mathbf{l} , which controls the pairwise spatial correlations. Intuitively, larger values of \mathbf{l} imply larger range of similarity in the vicinity. Formally, given a sampled point, we have a larger area of confidence in the vicinity, hence we should reach further to explore the uncertain area. That is why the generated sampling points in Fig. 3(b) are more spread out than in Fig. 3(a).

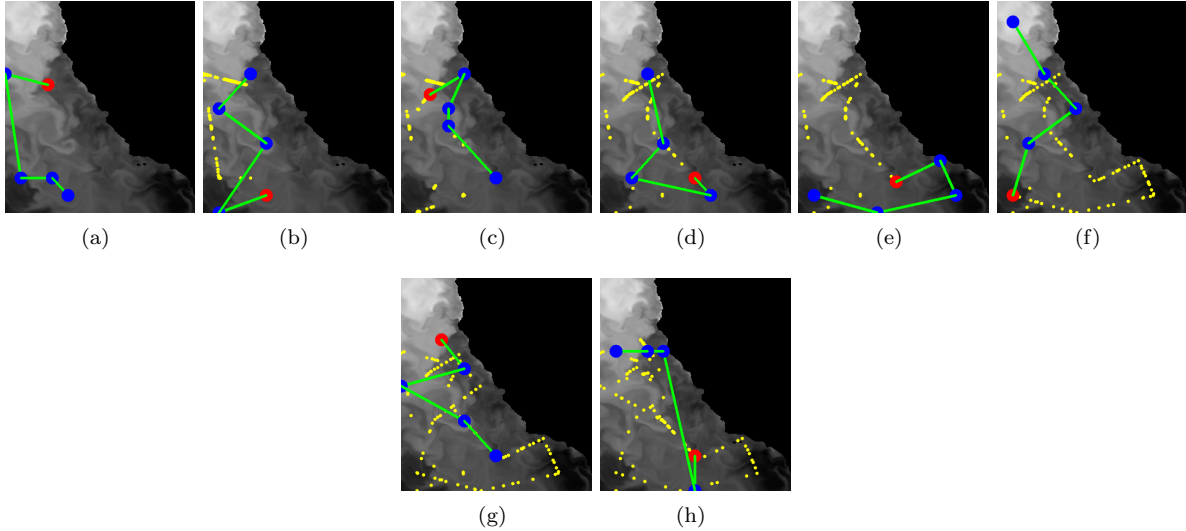


Figure 6: Sampling locations are generated randomly. Other routines are the same as those of Fig. 4.

Therefore, we use the data-driven planning and learning method to update the hyperparameters online. The process is demonstrated in Fig. 4, where a total number of 2000 sampling operations have been performed. Fig. 4(a) to 4(f) illustrate a series of snapshots at re-planning moments, so that all subsequent newly generated path segments can be fully seen. Specifically, a new informative path is computed if a previous path has been completed or the GP hyperparameters have been re-estimated and updated. For example, Figs. 4(b), 4(d), and 4(f) are the ones after completion of previous paths, whereas the remaining sub-figures are the ones generated after hyperparameter re-estimations. The red and blue points stand for the robot's current starting position and the informative sampling locations, respectively; the yellow dots represent the points stored in the SOGP BV-set. The robot launched from a random shore location (79, 236) and performed the sampling operations at each time step along the planned path. The initial hyperparameters are set to $\{\sigma_n^2 = \exp(-2), \sigma_f^2 = \exp(2), l_x = \exp(1), l_y = \exp(1)\}$ and we emulated the memory limit by setting the maximum size of the BV-set as $m = 100$. The threshold is set as $\rho_0 = 0.6$. The distribution patterns of the yellow dots in Fig. 4(a) to 4(f) reveal the sparseness of BV-set, indicating that as the robot gradually explores the whole map, the BV-set only stores those points that are the most useful for predicting the model. The corresponding predicted maps are shown in Fig. 5, from which we can see that the constructed models constantly converge to the ground truth and are able to characterize the general patterns of the environment at the final stages.

To evaluate the model prediction accuracy, we compare our method with an approach where the waypoints are generated randomly following uniform distribution. Note that the only difference between our method and the random approach is how the waypoints are generated. The other routines routines such as the online learning and the hyperparameter re-estimation remain the same. The reason is that as a high level planning component, the informative planner produces informative navigation waypoints which generally determine the scope of space explored and the amount of information acquired. For the random sampling approach, we also finish 2000 sampling operations so that the total number of sampling operations in the two methods are identical. The resultant sampling locations/paths and the retained samples from the random sampling approach are shown in Fig. 6; the corresponding predicted environmental models are shown in Fig. 7. Among all those snapshots, Fig. 7(b), 7(d), 7(f) and 7(h) are the ones generated after finishing previous paths, and the remaining ones are triggered after the updates of hyperparameters. Since the random sampling scheme does not take into account of informativeness, we can see that compared to the informative approach, the generated waypoints are less spread out for exploring those most unknown regions. The final predicted data map (model) is shown in Fig. 7(h), which clearly reveals that there is still a mis-prediction in the top area where it has never been explored at all.

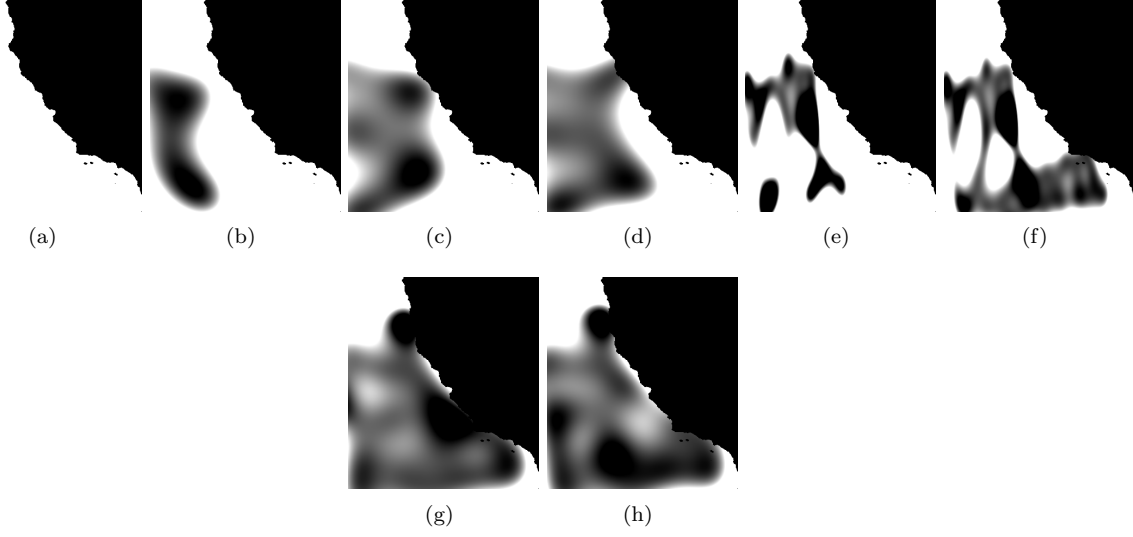


Figure 7: (a)-(h) The learned environment models from the random sampling approach. Each corresponds to a step in Fig. 6.

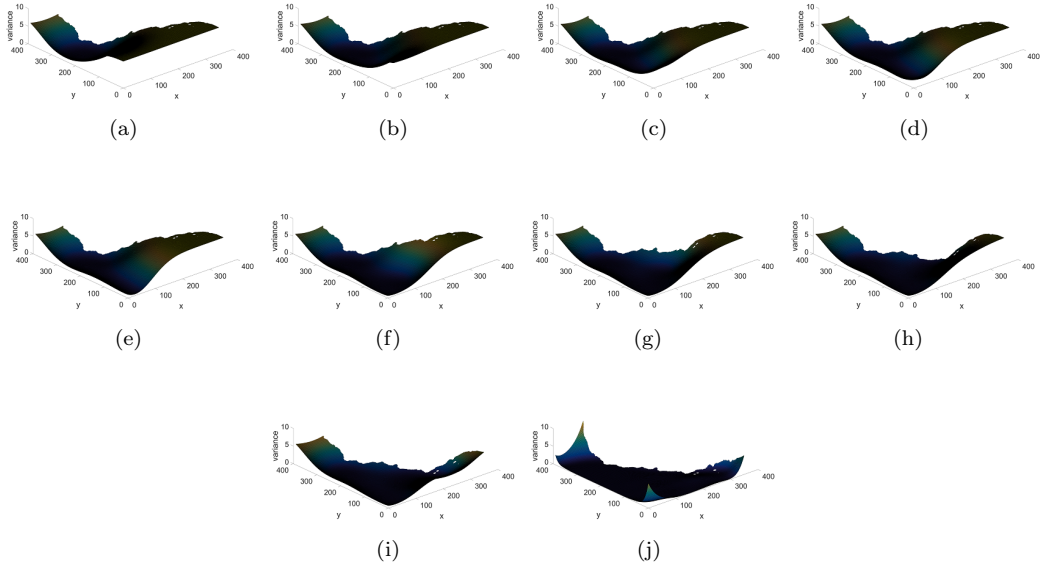


Figure 8: Maps of prediction variances from the proposed method. (a)-(i) Variances reduce as the robot follows planned path and collects data samples. (j) The final variance map that corresponds to the moments in Fig. 4(f) and Fig. 5(f).

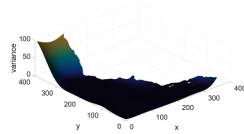


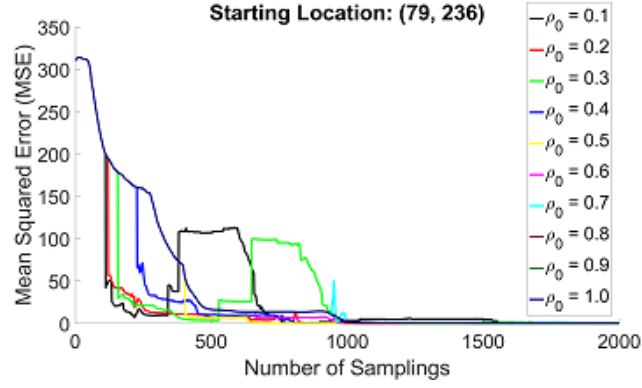
Figure 9: The final variance map of the random sampling approach. It corresponds to the moment in Fig. 6(h) and Fig. 7(h).

We also assess the model prediction variances. We create a variance map on which each “pixel” records the prediction variance of that point. Fig. 8 illustrates a series of variance maps along the sampling operations. We can see that the map gradually “falls towards the ground”, indicating a decrease of predication variances along the robot’s exploration. The variance map in the last time step is depicted in Fig. 8(j). Compared to Fig. 5, which is also the final stage variance map of the random approach, we can conclude that our proposed method succeeds in exploring most of the unknown and uncertain regions while the random scheme fails to do so.

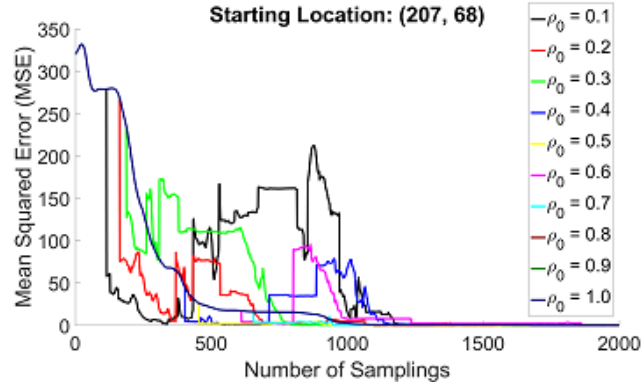
We then examine the prediction error between the predicted data map (i.e., GP’s predicted mean value) and the ground truth, as shown in Fig. 10. We use different thresholds ρ_0 and different launch locations to do the statistics. The x -axis represents the number of sampling operations, which is roughly proportional to the travel time (or distance). The y -axis is the MSE calculated using the whole map as a testing set. The figure reveals that, in general every setting follows a descending trend with the error being constantly reduced along the exploration process. By adjusting thresholds ρ_0 , we can see that there are more error fluctuations for low ρ_0 values. A possible reason is that, if the explored regions do not yet well cover the environment, the hyperparameter re-estimation might optimize only among some local regions rather than the entire map, causing a loss of generality and an overfitting problem. However, since the actual relation between two independent data points in the environment may change over time, a low ρ_0 value has the advantage of adapting to the changes more quickly because lower ρ_0 indicates that the hyperparameter re-estimations are done more frequently. The comparison between the proposed method and the random sampling approach is shown in Fig. 10(c), from which we can see that the error from our approach quickly decreases to a small value after 1000 samples are collected. Given long enough time, most of the environment can also be explored and covered using the random sampling approach, however, it takes more than 1500 sampling operations, and this number is much more than that our proposed approach. Note that there is still an error difference between the two methods at the final time step. Again, this is because in the random sampling approach the top area has never been explored.

Lastly, we investigate the scenario of dynamic environments, where there are variations along the temporal process. To construct such a spatiotemporal environment, we sequentially concatenate a series of “static environment frames”. In other words, we assume that during each small fixed-duration time interval, the environment is static. The environment varies among multiple time intervals. In greater detail, we change the underlying environment frames every 200 sampling operations. The variations of different salinity maps are shown in Fig. 11. In Fig. 12(a), we express the temporal variation by calculating the MSE between each frame/interval and the previous frame (red line) or the first frame (black line). Since the environment does not change within each interval, the curves are thus piecewise constant functions. We compare between the proposed framework and the random sampling scheme in the dynamic environment. The results are shown in Fig. 12, from which we can see that the error of our approach can still converge to a small value whereas there is a large error after 2000 sampling operations for the random sampling approach. The large error in the random sampling approach is due to the addition of temporal dynamics on top of the spatial variations. More specifically, a time-varying environment brings in more uncertainty than the static one. Since GP provides a measure of modelling uncertainty, or equivalently the informativeness, and since our informative planning framework can best acquire the informativeness, the collected data of which is then used to update the latent hyperparameters that support the GP, therefore, our method can well capture the spatiotemporal variations, leading to large error reduction. In contrast, the random sampling method does not take into account any uncertainty and so its exploration contains no informativeness. And this is why the random sampling strategy fails.

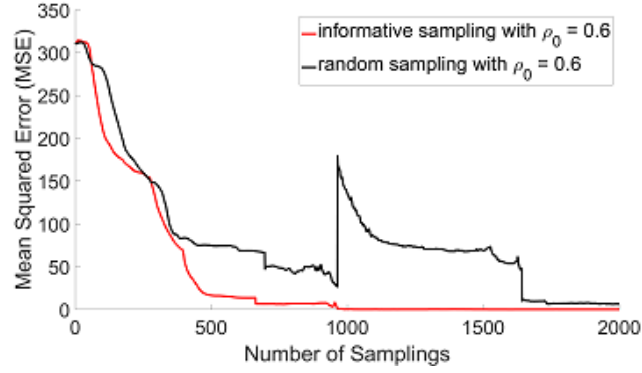
It is worth mentioning that, although our approach can quickly converge, the final error is larger than its counterpart application in the static environment. To show this, we zoom in the plot by focusing on the sampling operations from 1500 to 2000, as shown in Fig. 12(c). We can see that the error of the dynamic case is slightly higher than the static one by a value of 0.1. This makes sense as the time-varying environment has larger uncertainty than the static one. Note that there’s a sudden decrease of error at 1545, it’s because of the hyperparameter re-estimation.



(a)



(b)



(c)

Figure 10: (a)(b) The MSE results from trials of two different launching locations $\{(79, 236), (207, 68)\}$. In each case, the thresholds are tuned in a range $\rho_0 = \{0.1, 0.2, \dots, 1.0\}$. The y -axis is the MSE value while the x -axis is the number of sampling operations. (c) The MSE comparison between the informative paths and the random sampling paths with a fixed same threshold $\rho_0 = 0.6$.

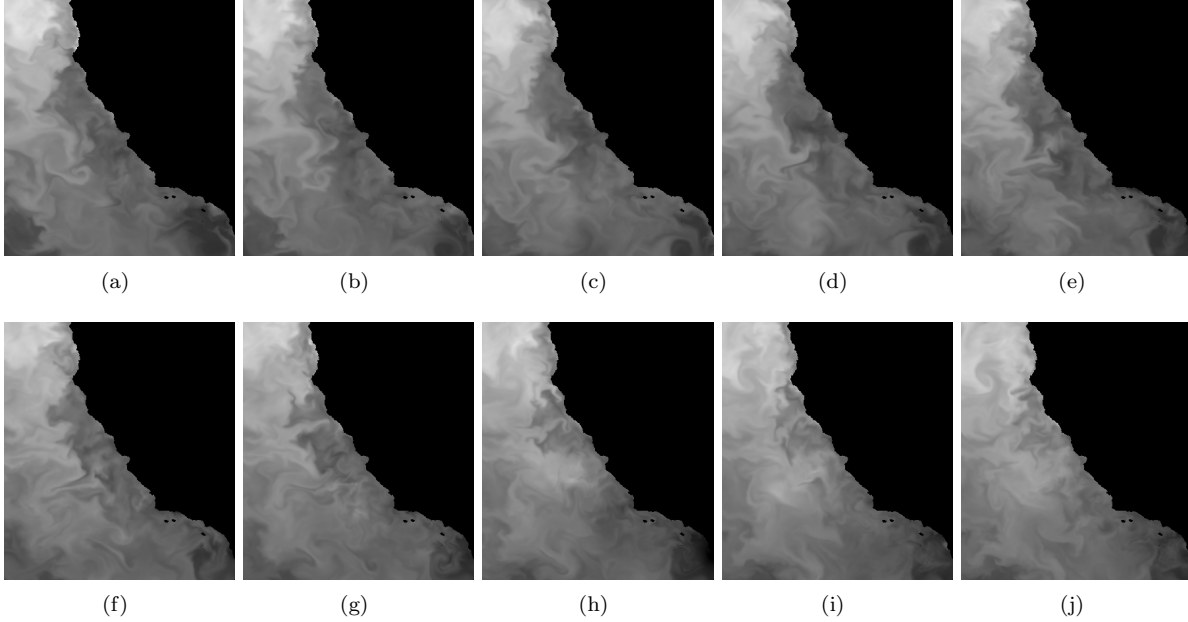


Figure 11: The spatiotemporal environment is constructed by concatenating a sequence of groundtruth salinity maps at different times.

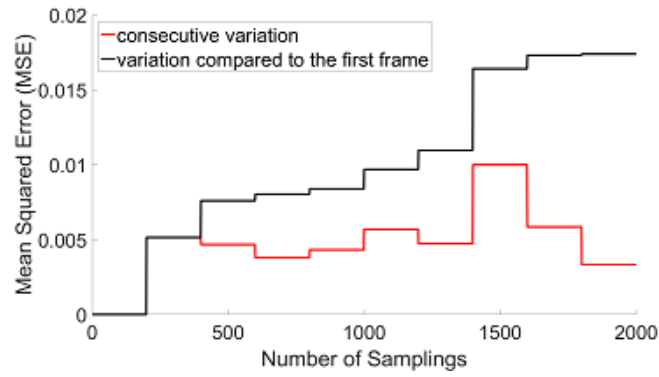
6 Field Trials with an Autonomous Surface Vehicle

Field trials were conducted with an autonomous surface vehicle (ASV) developed at the University of Southern California (USC), pictured in Fig. 13(a), at Puddingstone Reservoir in San Dimas, California. The USC ASV is around 2 *m* long and 0.8 *m* wide, actuated by two electrical thrusters and a rudder. It is equipped with an IMU and RTK GPS for localization, a sonar pinger to measure depth and onboard computers for control. In addition it has sensors, such as forward looking sonar, laser range finder and stereo cameras, that were not used in this trial.

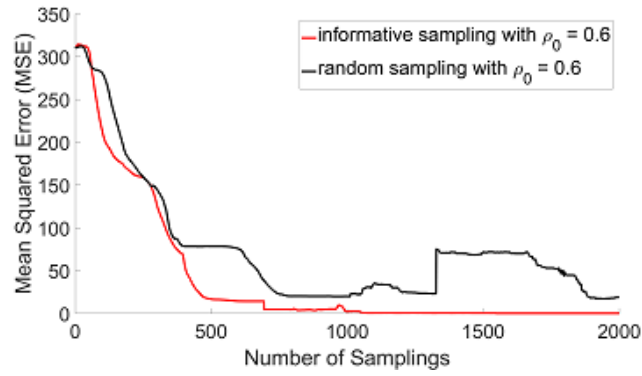
In the field trial, we focus on sampling in the static environment. There are two reasons for doing so. The first reason lies in the ease of obtaining ground truth, with which we can perform comparisons and evaluations. In contrast, in a time-varying environment this is almost impossible unless we use many vehicles simultaneously where those vehicles are mainly used for constructing ground-truth. The second reason lies in that, as we analyzed in the simulation section, a time-varying environment can be represented by a sequence of static environments. One can regard the static environment as a static interval of a long-term time-varying environment.

The field trial site is shown in Fig. 13(b). Note that most inland lakes and reservoirs contain not only little temporal variation but also little spatial variation, for most environment attributes such as temperature, salinity, pollution or nutrition contents, etc. These physical or chemical attributes are easily spread along with the dynamic water flow and they are typically uniform across the entire environment. Therefore, we opt to model the *water depth* from the water surface to the bottom of the lake. The water depth gives us enough spatial variation and it stays constant during and between runs so we can easily collect ground truth for comparison.

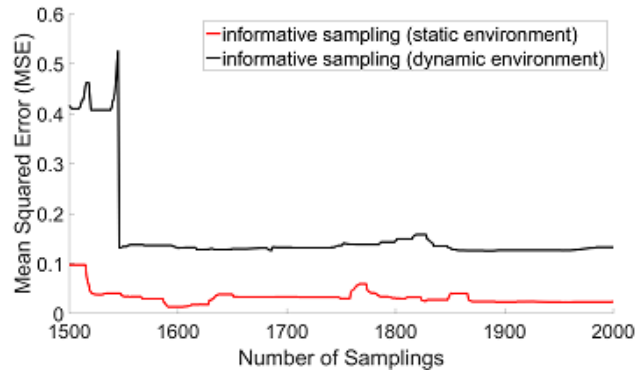
The ground truth was obtained by combining water depth data from multiple prior runs of our YSI EcoMapper AUV, as shown in Fig. 14(a). The ASV was equipped with a Furuno 235DST-PWE depth pinger which has a range of 100 *m*, beam width of 7° and sampling rate of 1 Hz. We collected the depth data using the comprehensive Lawn Mower approach, resulting in a dense depth map for the experimental area, as shown



(a)



(b)



(c)

Figure 12: (a) The salinity variation measured in MSE. The red curve is calculated by comparing each current frame with its previous one, while the black one is by comparing each current frame with the first frame; (b) MSE comparison between the informative sampling method and the random sampling scheme in a dynamic environment; (c) MSE comparison between the static and dynamic environments, for the informative sampling method upon convergence.

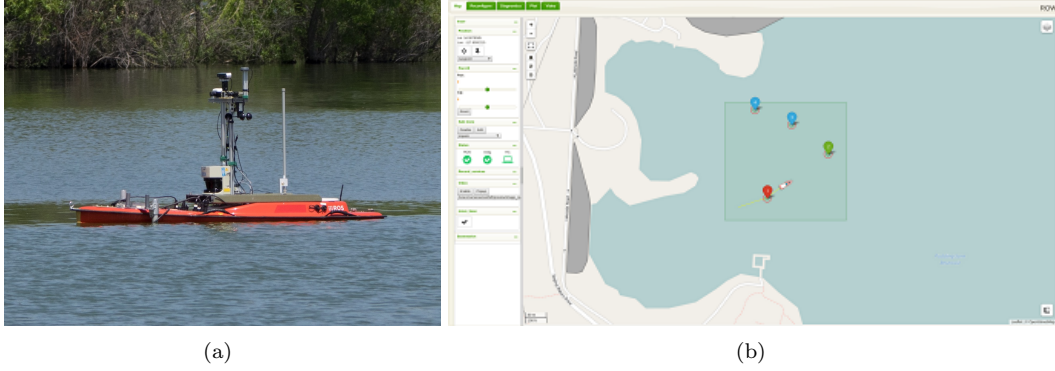


Figure 13: (a) The ASV developed at the University of Southern California; (b) The planning interface of the ASV. The bounding box pre-defines a restricted mission area and the numbered pinpoints are the waypoints that the ASV is going to traverse.

in Fig. 14(b). In these figures, the color tone corresponds to the water depth where warmer color represents deeper depth while colder color represents shallower depth. The maximum depth is roughly 19 meters.

In our experiment, in order to shorten the duration of each trial and exclude the possibility of bumping into obstacles, we intentionally define the experimental area to be a sub-region of the groundtruth. It is set to be a square ranging from latitude 34.0875 to 34.0890 and longitude -117.8082 to -117.8102 . The size is about 170×165 squared meters. The boat is restricted to run within this predefined area for about 20 minutes per one trial. The initial starting location is set to be roughly at the midpoint of the area. The initial starting location can actually be arbitrary since we assume no prior samples in advance. That is, we have no idea about the depth at all before we start collecting the data. However if we have some prior samples (e.g. collected from previous trials or from some static sensors), we might tend to set the initial starting location to be away from the area where prior samples are located. In this way, some traveling cost can be saved by not passing the known area.

Next, we describe how the ASV is controlled. Here we skip the low level control and only discuss the high level control in detail. The high level controller of the ASV is used by specifying a sequence of waypoints for it to traverse. Fig. 13(b) shows the interface of our ASV and the example of controlling it through waypoints specification. The bounding box is exactly the predefined area described in the previous paragraph, and the pinpoints are the waypoints that the ASV is going to traversed. The number on the pinpoints denotes the traversal sequence. Due to the wind disturbance or the imprecise localization (in case we lose RTK lock), it is nearly impossible to arrive at the exact location of the each waypoint. Therefore, we set a bounding circle with radius of 3 meters for each waypoint such that it is deemed to be reached when the relative distance between the ASV and the waypoint is less than the radius. The sampling rate of the depth pinger is 1 Hz, which is the same as the one used for collecting the ground truth data. We also set the speed of the ASV to be a constant of 1.5 m/s throughout the entire trial. Because the ASV is always running in a constant speed (excluding the effect of the wind), we can assume that the distance traveled or the number of sampling operations are equivalent to the meaning of duration. The maximum size of the BV-set and the threshold are also set to $m = 100$ and $\rho_0 = 0.6$, as in the simulation with ocean data. The initial hyperparameters of the covariance function are empirically set to $\{\sigma_n^2 = \exp(-2), \sigma_f^2 = \exp(4), l_x = \exp(-8), l_y = \exp(-8)\}$. Here the l hyperparameters are significantly smaller than the ones in Sec.5. It is because we are now interested in the depth structure in the reservoir rather than the salinity in the ocean. It is the different properties between the two interested targets that cause different settings of hyperparameters.

In the field trials, we also compare our approach with the random approach described in Sec.5. First we compare the recorded paths resulted from the random approach and our approach in Fig. 15. The \times denotes the starting location of the boat, which is roughly at the center of the whole area. Fig. 15(a) shows the

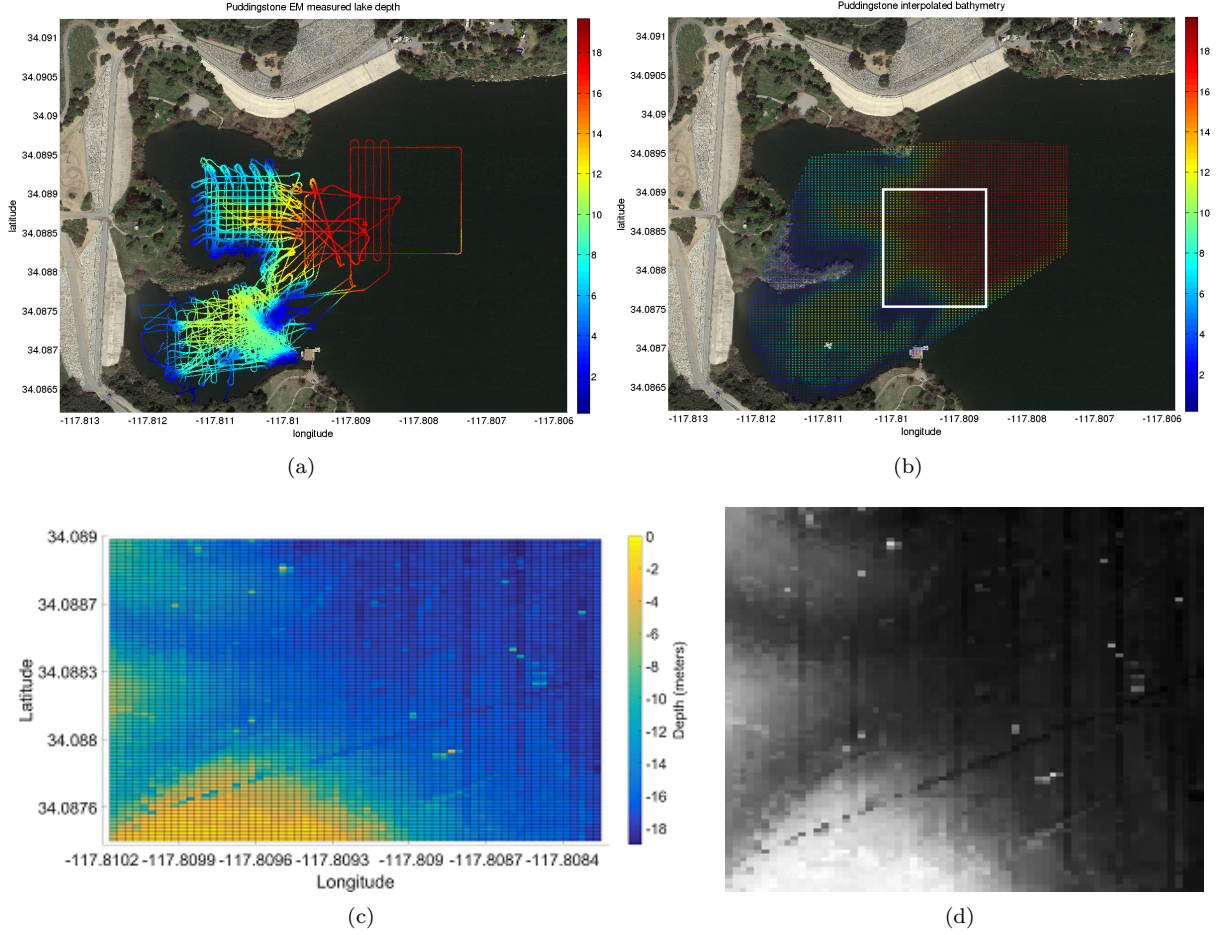


Figure 14: (a) Lawn Mower paths for estimating the ground truth; (b) Constructed ground truth map. The x -axis is the longitude while the y -axis is the latitude. The depth unit is in meters; (c) A restricted bounding-box area is used for our field trial. This area corresponds to the white box in Fig.(b) (d) The same groundtruth area in gray-scale. The darker, the deeper.

paths resulted from uniformly selecting random waypoints for the boat to traverse while Fig. 15(b) shows the result of our proposed approach. Given the same constant speed of the boat and the time, they both collected the same amount of data and travelled the same distance. One might notice the boat is wandering at some certain points, this is due to the boat is recalculating a new path since it has either completed the current batch of waypoints or the hyperparameter re-estimation procedure has been triggered. During recalculation, we intentionally let the ASV park and disengage its motors at the current location and stop sampling. Because there's wind blowing the ASV away from the parking location in the field and the ASV would try to go back to the parking location, as a result, it seems wandering around the parking location. In Fig. 15(a), we can see there are some areas missed to be explored by the boat, for example, the right lower and left parts, while in Fig. 15(b), most of the area is already covered in the same amount of time. The results shows that our approach is better in terms of coverage given the same amount of time.

Next we compare the performance in terms of modeling the environment. Fig. 16 shows the final depth map estimated from the SOGP using samples collected from the random paths. Fig. 16(b) also shows the locations of preserved samples stored in the SOGP BV-set. We can see that in the areas missed by the boat, the depth estimate is inaccurate because few samples in the missed areas are stored. However for the area where samples are stored, the SOGP can captures the depth structure fairly well, for example, the top-right part is relatively deep while the bottom-left part is shallow.



Figure 15: Recorded paths in the field trial. The ‘x’ is the starting location. (a) The paths from the random sampling method; (b) The paths from this proposed method.

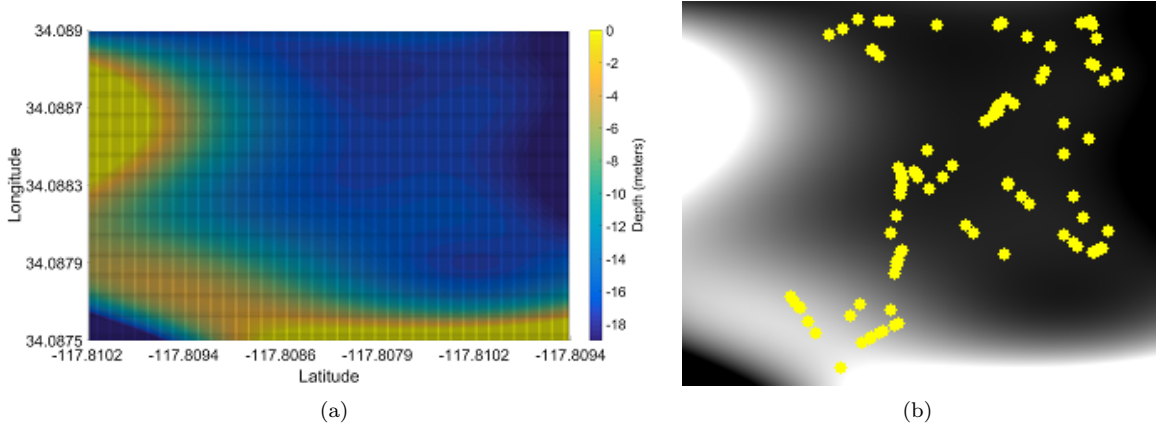


Figure 16: (a) The depth estimate result from the random sampling approach; (b) The same depth estimate in gray-scale. The yellow dots denote the data points stored in the SOGP BV-set.

Fig. 17 shows the progress of the estimated depth map along the informative paths proposed by us. The interval between each depth map is 150 sampling operations and Fig. 17(i) is the final result. Initially we have no idea about the field, the depth in most of the region are estimated as 0, but as it progresses, we are getting better understanding about the overall depth structure. From the final result, we can see that it captures the distribution of depth a lot better than the random approach. It successfully models the three shallow regions on the left and the deep region on the right. From Fig. 17(j), we can also see that the samples stored in the BV-set are distributed to describe the depth structure accordingly.

Lastly, the corresponding numerical estimation errors are also compared. Fig. 18 shows the MSE plot of the two approaches calculated from comparing with the ground truth data. The y -axis is the MSE value while the x -axis is the total number of sampling operations. The overall error in both approaches are decreasing as time goes although there are some significant fluctuations and peaks (e.g. there is a peak at $x = 300$ in random approach). Again, it is possibly because the regions are not yet well covered, the hyperparameter re-estimation procedure might optimize only among some local regions rather than the entire map. An obvious difference between the two approaches is that the final error of our approach has already converged down to 1.4035 in the end while the other is still relatively high. This indicates that the informative planning in our approach can significantly reduce the cost of exploration and we believe that with larger areas defined, the reduction should be more remarkable.

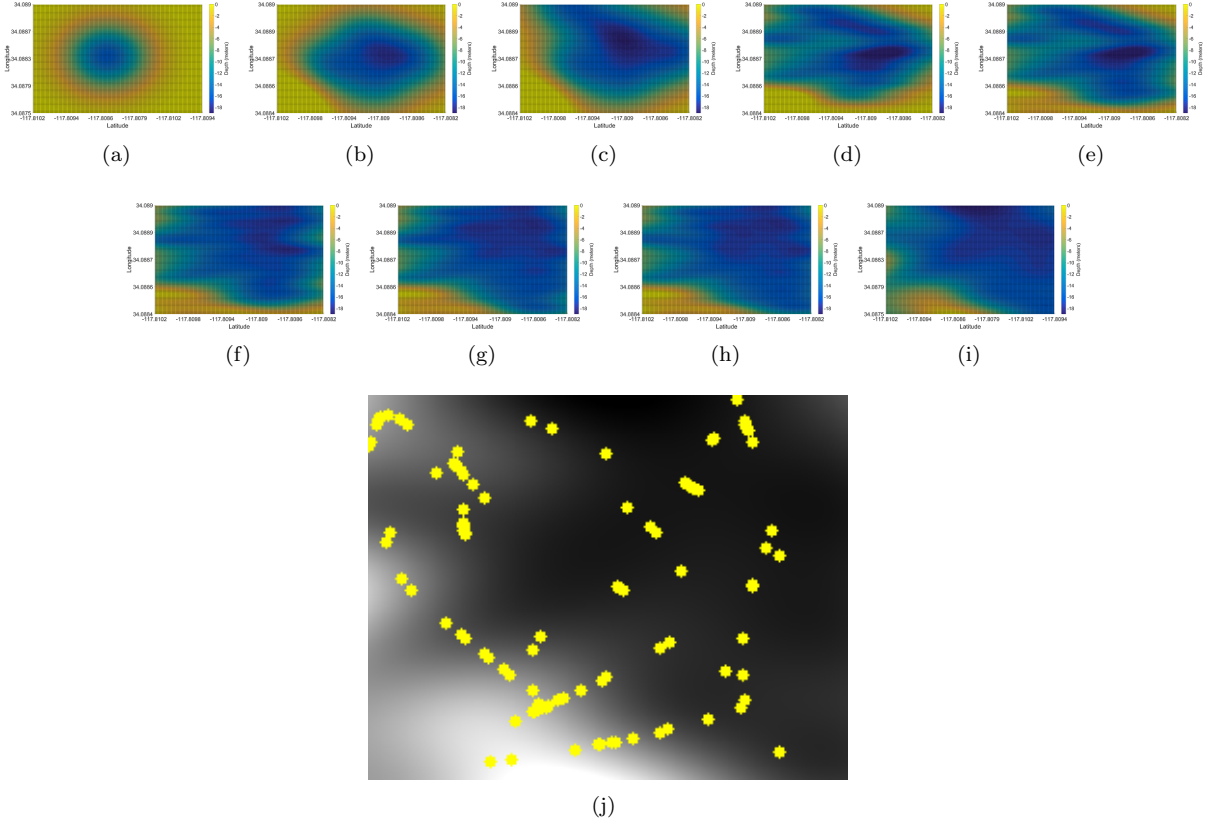


Figure 17: (a)-(i) The progress of depth-map estimation along the informative sampling paths; (j) The final estimated depth map, which also corresponds to state (i). The retained data points (yellow dots) have a good coverage of the environment.

7 Conclusions

Environmental monitoring entails persistent presence by robots. This suggests that both planning and learning are likely to constitute critical components of any robotic system built for monitoring. In this paper, we present an informative planning and online learning method that enables an autonomous aquatic vehicle to effectively perform persistent ocean monitoring tasks. Our proposed framework iterates between a planning component that is designed to collect data with the richest information content, and a sparse Gaussian Process learning component where the environmental model and hyperparameters are learned online by selecting and utilizing only a subset of data that makes the greatest contribution. We have conducted both simulations and field trials by comparing with a non-informative sampling method, and the results show that our method produces a good match between the predicted models and the ground truths, with superior decrements for both prediction errors and map variances.

References

- Anderson, K. and Gaston, K. J. (2013). Lightweight unmanned aerial vehicles will revolutionize spatial ecology. *Frontiers in Ecology and the Environment*, 11(3):138–146.
- Bai, S., Wang, J., Chen, F., and Englot, B. (2016). Information-theoretic exploration with bayesian optimization. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*, pages 1816–1822.

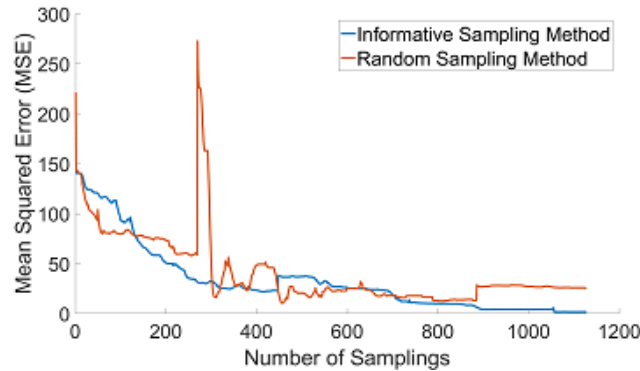


Figure 18: The MSE comparison between the informative sampling approach and the random sampling method in the field trial. The y -axis is the MSE value while the x -axis is the number of sampling operations.

- Binney, J., Krause, A., and Sukhatme, G. S. (2010). Informative path planning for an autonomous underwater vehicle. In *International Conference on Robotics and Automation*, pages 4791–4796.
- Binney, J., Krause, A., and Sukhatme, G. S. (2013). Optimizing waypoints for monitoring spatiotemporal phenomena. *International Journal on Robotics Research (IJRR)*, 32(8):873–888.
- Blum, M. and Riedmiller, M. A. (2013). Optimization of gaussian process hyperparameters using rprop. In *ESANN*.
- Brooks, A., Makarenko, A., and Upcroft, B. (2008). Gaussian process models for indoor and outdoor sensor-centric robot localization. *IEEE Transactions on Robotics*, 24(6):1341–1351.
- Cao, N., Low, K. H., and Dolan, J. M. (2013). Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, pages 7–14.
- Cristianini, N., Elisseeff, A., Shawe-Taylor, J., and Kandola, J. (2001). On kernel-target alignment. *Advances in neural information processing systems*.
- Csató, L. and Oppor, M. (2002). Sparse on-line gaussian processes. *Neural computation*, 14(3):641–668.
- Dunbabin, M. and Marques, L. (2012). Robotics for environmental monitoring: Significant advancements and applications. *IEEE Robot. Autom. Mag.*, 19(1):24 – 39.
- Erickson, P., Cline, M., Tirpankar, N., and Henderson, T. (2015). Gaussian processes for multi-sensor environmental monitoring. In *2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 208–213.
- Farrell, J. A., Pang, S., and Li, W. (2005). Chemical plume tracing via an autonomous underwater vehicle. *IEEE Journal of Oceanic Engineering*, 30(2):428–442.
- Fiorelli, E., Bhatta, P., and Leonard, N. E. (2003). Adaptive sampling using feedback control of an autonomous underwater glider fleet. In *Proc. 13th Int. Symposium on Unmanned Untethered Submersible Tech*, pages 1–16.
- Guestrin, C., Krause, A., and Singh, A. P. (2005). Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 265–272. ACM.
- Hajieghrary, H., Tom, A. F., Hsieh, M. A., et al. (2015). An information theoretic source seeking strategy for plume tracking in 3d turbulent fields. In *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–8. IEEE.

- Hollinger, G. A., Pereira, A. A., Binney, J., Somers, T., and Sukhatme, G. S. (2016). Learning uncertainty in ocean current predictions for safe and reliable navigation of underwater vehicles. *Journal of Field Robotics*, 33(1):47–66.
- Hollinger, G. A., Pereira, A. A., and Sukhatme, G. S. (2013). Learning uncertainty models for reliable operation of autonomous underwater vehicles. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5593–5599. IEEE.
- Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231 – 247.
- Leonard, N. E., Paley, D. A., Davis, R. E., Fratantoni, D. M., Lekien, F., and Zhang, F. (2010). Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in monterey bay. *Journal of Field Robotics*, 27(6):718–740.
- Liu, X., Xi, T., and Ngai, E. (2016). Data modelling with gaussian process in sensor networks for urban environmental monitoring. In *Proc. 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 457–462. IEEE Computer Society.
- Lloret, J., Garcia, M., Bri, D., and Sendra, S. (2009). A wireless sensor network deployment for rural and forest fire detection and verification. *sensors*, 9(11):8722–8747.
- Low, K. H. (2009). *Multi-robot Adaptive Exploration and Mapping for Environmental Sensing Applications*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA.
- Low, K. H., Dolan, J. M., and Khosla, P. (2011). Active markov information-theoretic path planning for robotic environmental sensing. In *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS-11)*, pages 753–760.
- Ma, K.-C., Liu, L., and Sukhatme, G. S. (2016a). An information-driven and disturbance-aware planning method for long-term ocean monitoring. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- Ma, K.-C., Liu, L., and Sukhatme, G. S. (2016b). Multi-robot informative and adaptive planning for persistent environmental monitoring. In *International Symposium on Distributed Autonomous Robotic Systems (DARS)*.
- Ma, K.-C., Liu, L., and Sukhatme, G. S. (2017). Informative planning and online learning with sparse gaussian processes. In *IEEE International Conference on Robotics and Automation (ICRA)*.
- Meliou, A., Krause, A., Guestrin, C., and Hellerstein, J. M. (2007). Nonmyopic informative path planning in spatio-temporal models. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 602–607.
- Miles, T., Lee, S. H., Whlin, A., Ha, H. K., Kim, T. W., Assmann, K. M., and Schofield, O. (2015). Glider observations of the dotson ice shelf outflow. *Deep Sea Research Part II: Topical Studies in Oceanography*.
- Neal, R. M. (1996). Bayesian learning for neural networks.
- Nguyen-tuong, D. and Peters, J. (2008). Local gaussian process regression for real time online model learning and control. In *In Advances in Neural Information Processing Systems 22 (NIPS)*.
- Nuske, S., Achar, S., Bates, T., Narasimhan, S., and Singh, S. (2011). Yield estimation in vineyards by visual grape detection. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2352–2358. IEEE.
- Oliveira, L. M. and Rodrigues, J. J. (2011). Wireless sensor networks: a survey on environmental monitoring. *Journal of Communications*, 6:143–151.

- Opper, M. (1998). On-line learning in neural networks. chapter A Bayesian Approach to On-line Learning, pages 363–378. Cambridge University Press, New York, NY, USA.
- Ouyang, R., Low, K. H., Chen, J., and Jaillet, P. (2014). Multi-robot active sensing of non-stationary gaussian process-based environmental phenomena. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pages 573–580.
- Paley, D. A., Zhang, F., Fratantoni, D. M., and Leonard, N. E. (2008). Glider control for ocean sampling: The glider coordinated control system. *IEEE Transaction on Control System Technology*, 16(4):735–744.
- Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1996). *Numerical recipes in C*, volume 2. Cambridge university press Cambridge.
- Ranganathan, A., Yang, M.-H., and Ho, J. (2011). Online Sparse Gaussian Process Regression and Its Applications. *IEEE Transactions on Image Processing*, 20(2):391–404.
- Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press.
- Shchepetkin, A. F. and McWilliams, J. C. (2005). The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean Modelling*, 9(4):347–404.
- Singh, A., Krause, A., Guestrin, C., Kaiser, W., and Batalin, M. (2007). Efficient planning of informative paths for multiple robots. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI’07*, pages 2204–2211.
- Smith, R. N., Schwager, M., Smith, S. L., Jones, B. H., Rus, D., and Sukhatme, G. S. (2011). Persistent ocean monitoring with underwater gliders: Adapting sampling resolution. *Journal of Field Robotics*, 28(5):714–741.
- Soltero, D. E., Schwager, M., and Rus, D. (2012). Generating informative paths for persistent sensing in unknown environments. In *IROS*, pages 2172–2179.
- Tokekar, P., Bhadauria, D., Studenski, A., and Isler, V. (2010). A robotic system for monitoring carp in minnesota lakes. *Journal of Field Robotics*, 27(6):779–789.
- Trincavelli, M., Reggente, M., Coradeschi, S., Loutfi, A., Ishida, H., and Lilienthal, A. J. (2008). Towards environmental monitoring with mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2210–2215.
- Watts, A. C., Ambrosia, V. G., and Hinkley, E. A. (2012). Unmanned Aircraft Systems in Remote Sensing and Scientific Research: Classification and Considerations of Use. *Remote Sensing*, 4(6):1671–1692.
- Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., and Welsh, M. (2006). Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 381–396. USENIX Association.
- Yang, C. (2012). A high-resolution airborne four-camera imaging system for agricultural remote sensing. *Computers and electronics in agriculture*, 88:13–24.
- Yu, J., Schwager, M., and Rus, D. (2014). Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.